The Business Case for Better Software Practices

www.construx.com
Copyright Notice

These class materials are © 2002-2008 Construx Software Builders, Inc.

All Rights Reserved. No part of the contents of this seminar may be reproduced or transmitted in any form or by any means without the written permission of Construx Software Builders, Inc.
State of the Practice
Disappointing Project Outcomes

- Average schedule overrun may be as high as 100%
- About one-quarter of all projects are cancelled
- Reasons for these outcomes are complicated
Disappointing Project Outcomes

- **Actual inefficiencies**
  - Most projects are run somewhat inefficiently
  - Average developer reads less than 1 professional book/year and subscribes to no professional journals

- **Perceived inefficiencies**
  - Management & customer expectations are often unrealistic and unachievable
  - Some management and customer actions actually undermine effective project performance
Average Practice is Close to the Worst Practice

% of Org’s

Effectiveness

Mean

Expected Distribution

Actual Distribution
Lack of Understanding of Project Scaling Issues

Cost Breakdown on a Small Project (2KLOC)
- Unit Testing: 23%
- Integration: 12%
- System Testing: 12%
- Coding & Debugging: 22%
- Detailed Design: 17%
- Architecture: 14%

Cost Breakdown on a Large Project (500 KLOC)
- Unit Testing: 8%
- Integration: 21%
- System Testing: 19%
- Coding & Debugging: 8%
- Detailed Design: 18%
- Architecture: 26%

Construction = 2/3 of Effort
Construction = 1/3 of Effort
1. How much are you spending on software?
2. How confident are you that your “buy” decisions should not be “build” decisions?
3. What percentage of your costs arise from unplanned rework?
4. What percentage of your projects are on time and on budget?
5. How confident are you that your current projects will perform to their estimates?
6. What percentage of your current projects are most likely to be cancelled?

7. How do your teams’ skills compare to industry averages?

8. How do the capabilities of your organization compare to other, similar organizations?

9. How satisfied (quantitatively) are users of your software?

10. How much (quantitatively) has your productivity improved in the past 12 months?
Improved Software Practices are Business’s Last Great Frontier
### ROI for Selected Practices

<table>
<thead>
<tr>
<th>Practice</th>
<th>12-month ROI</th>
<th>36-month ROI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal code inspections</td>
<td>250%</td>
<td>1200%</td>
</tr>
<tr>
<td>Formal design inspections</td>
<td>350%</td>
<td>1000%</td>
</tr>
<tr>
<td>Cost and quality estimation tools</td>
<td>250%</td>
<td>1200%</td>
</tr>
<tr>
<td>Long-range technology planning</td>
<td>100%</td>
<td>1000%</td>
</tr>
<tr>
<td>Productivity measurements</td>
<td>150%</td>
<td>600%</td>
</tr>
<tr>
<td>Process assessments</td>
<td>150%</td>
<td>600%</td>
</tr>
<tr>
<td>Management training</td>
<td>115%</td>
<td>550%</td>
</tr>
<tr>
<td>Technical staff training</td>
<td>90%</td>
<td>500%</td>
</tr>
</tbody>
</table>

Improved software practices pay an average ROI of 500% (including false starts), and continued improvement is sustainable for many years.

The best organizations have sustained ROIs of 900% from software improvement initiatives for many years.

Where Does the “R” Come From?
Where Costs Come From:

Lifecycle Cost Profile

Typical Project (Pathological Project)
Advanced Project (Healthy Project)
Where Costs Come From:

Lifecycle Cost Profile (cont.)

- Requirements Architecture Detailed Design
- Construction Testing and Debugging

Relative Effort

- Typical Project (Pathological Project)
- Advanced Project (Healthy Project)
- Unplanned rework
- Process overhead

Construx
Where Costs Come From:

Activity Breakdown

Cost Breakdown for an Average Project:
- Rework: 60%
- Planned Work: 40%

Cost Breakdown for an Expertly-Run Project:
- Rework: 20%
- Planned Work: 80%
Reduced Cost

- Improving software practices reduce costs an average of ~35% per year
- The improvement is sustainable for several years, and the potential is much higher
- The best organizations have sustained cost improvements of 55%+ per year

Better Quality

- Poor quality is the single largest cost driver for most projects
- Improved software practices improve quality an average of ~40% per year
- This improvement is sustainable for several years, and the potential is much higher
- The best organizations have sustained quality improvements of 70%+ per year

Improved Cycle Time

- Improved software practices shorten schedules an average of ~15-20% per year
- This improvement is sustainable for several years, and the potential is much higher
- The best organizations have sustained schedule improvements of ~20-25% per year

Better Predictability

Project Performance Compared to Estimate

Actual Results as a Percentage of Estimated Results

Predictability before improvement

Predictability after improvement
Enhanced Morale

Percentage of Employees Who Rate Their Own Morale as "Good" or "Excellent"

## ROI Examples

<table>
<thead>
<tr>
<th>Organization</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boeing Information Systems</td>
<td>Estimates within 20%, $5.5 million saved in 1 year</td>
</tr>
<tr>
<td>BDN</td>
<td>ROI 300%</td>
</tr>
<tr>
<td>CSC</td>
<td>65% reduction in error rates</td>
</tr>
<tr>
<td>Harris ISD DPL</td>
<td>90% defect rate reduction; 2.5x productivity gain</td>
</tr>
<tr>
<td>Hewlett-Packard SESD</td>
<td>ROI 900%</td>
</tr>
<tr>
<td>Hughes</td>
<td>$2 million annual reduction in cost overruns</td>
</tr>
</tbody>
</table>
### ROI Examples

<table>
<thead>
<tr>
<th>Organization</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBM Toronto</td>
<td>90% reduction in delivered defects, 80% reduction in rework</td>
</tr>
<tr>
<td>Motorola GED</td>
<td>2-3X productivity improvement, 2-7X cycle time reduction, ROI 677%</td>
</tr>
<tr>
<td>Philips</td>
<td>ROI 750%</td>
</tr>
<tr>
<td>Raytheon</td>
<td>ROI 770%</td>
</tr>
<tr>
<td>Siemens</td>
<td>90% reduction in released defects</td>
</tr>
<tr>
<td>Schlumberger</td>
<td>4X reduction in beta test bugs</td>
</tr>
</tbody>
</table>
ROI Examples

<table>
<thead>
<tr>
<th>Organization</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Telcordia</td>
<td>Defects 1/10 industry average, customer satisfaction increased from 60-91% over 4 years</td>
</tr>
<tr>
<td>Texas Instruments – Systems Group</td>
<td>90% reduction in delivered defects</td>
</tr>
<tr>
<td>Thomson CSF</td>
<td>ROI 360%</td>
</tr>
<tr>
<td>US Navy</td>
<td>ROI 410%</td>
</tr>
<tr>
<td>USAF Ogden Air Logistics Center</td>
<td>ROI 1900%</td>
</tr>
<tr>
<td>USAF Oklahoma City Air Logistics Center</td>
<td>ROI 635%</td>
</tr>
<tr>
<td>USAF Tinker Air Force Base</td>
<td>ROI 600%</td>
</tr>
</tbody>
</table>
Other Benefits

- Direct ROI is from better operational efficiency
- Indirect ROI may be greater
  - Predictability--product launch, changes in business practices, etc.
  - Inter-group coordination
  - Cost control
  - Risk Reduction
ROI Implications

- Size of investment required varies
- Payback periods vary
- Not all investments are possible initially—some larger ROIs are dependent on previous investments with smaller ROI
- Best starting point depends on organizational specifics
What Prevents Companies from Seizing This Opportunity?
Procrastination:
“Hard work often pays off over time, but laziness always pays off now”
Most Software Professionals Have Not Seen Software Development at its Best
What Prevents Companies from Seizing the Opportunity?

- Many companies are seizing the opportunity!
- Successful small projects cause complacency—leading to unsuccessful large projects
- Too much time spent fighting current fires to prevent future fires
- Improving operational efficiency is not a priority during gold rushes
- Haven’t seen the compelling business case for better software practices!
Strategy
Cost Drivers

- Product Complexity
- Analyst Capability (general)
- Programmer Capability (general)
- Time Constraint
- Personnel Continuity
- Multi-site Development
- Required Software Reliability
- Documentation Match to Lifecycle Needs
- Applications Experience
- Use of Software Tools
- Platform Volatility
- Storage Constraint
- Process Maturity
- Language and Tools Experience
- Database Size
- Platform Experience
- Architecture and Risk Resolution
- Precedentedness
- Developed for Reuse
- Team Cohesion
- Development Flexibility

Cost Drivers

- Product Complexity: 2.38
- Analyst Capability (general): 2.00
- Programmer Capability (general): 1.76
- Time Constraint: 1.63
- Personnel Continuity: 1.59
- Multi-site Development: 1.56
- Required Software Reliability: 1.54
- Documentation Match to Lifecycle Needs: 1.52
- Applications Experience: 1.51
- Use of Software Tools: 1.50
- Platform Volatility: 1.49
- Storage Constraint: 1.46
- Process Maturity: 1.43
- Language and Tools Experience: 1.43
- Database Size: 1.42
- Platform Experience: 1.40
- Architecture and Risk Resolution: 1.38
- Precedentedness: 1.33
- Developed for Reuse: 1.31
- Team Cohesion: 1.29
- Development Flexibility: 1.26
Organizational Focus

- Few factors are readily within the control of a single project
- Few factors are totally outside the control of both the project and the organization
- Leverage for improved software practices is mostly at the organizational level rather than the project level
- Construx’s focus is on identifying best focus areas and then supporting successful improvements within those areas
Focus on Low Hanging Fruit

- Lots of proven practices are available
- Risk of not using these practices is substantially higher than of using them
<table>
<thead>
<tr>
<th>Project planning and management practices</th>
<th>Requirements engineering practices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automated estimation tools (1973)</td>
<td>Change board (1978)</td>
</tr>
<tr>
<td>Measurement (1977)</td>
<td>JAD sessions (1985)</td>
</tr>
<tr>
<td>Productivity environments (1984)</td>
<td></td>
</tr>
<tr>
<td>Risk-management planning (1981)</td>
<td></td>
</tr>
</tbody>
</table>
Low Hanging Fruit (cont.)

- Design practices
  - Information hiding (1972)
  - Design for change (1979)
- Construction practices
  - Source code control (1980)
  - Incremental integration (1979)
- Quality assurance practices
  - Branch-coverage testing (1979)
  - Inspections (1976)
- Process improvement
  - SW-CMM (1987)
  - Software Engineering Process Groups (1988?)
Where to Start

❖ Generalities:
- Requirements, project planning, project tracking, quality assurance, configuration management subcontractor management
- These are the SEI CMM-SW’s Level 2 KPAs
- Construx’s consulting experience bears this out as a generality

❖ Specifics vary greatly
Scoping the Investment Required
Schedule Required

Number of months to move to the next CMM Level

- Level 1 to 2: 76 Orgs (26 months)
- Level 2 to 3: 56 Orgs (19 months)

- Largest observed value that is not an outlier
- 75th Percentile
- Median
- 25th Percentile
- Smallest observed value that is not an outlier
The average investment is 2.0% of fully burdened labor costs, i.e., $2000-$3000 per employee per year
This investment includes the direct costs of training, coaching, and facilities improvements

References


Lawrence H. Putnam, “Linking the QSM Productivity Index with the SEI Maturity Level.”


Construx Software is committed to helping individuals and organizations improve their software development practices. For information about our training and consulting services, contact stevemcc@construx.com.

Seminar Schedule: www.construx.com/calendar

Construx
10900 NE 8th Street, Suite 1350
Bellevue, WA 98004
+1 (866) 296-6300
www.construx.com